Meta-learning how to forecast time series

Thiyanga S Talagala
Department of Statistics, Faculty of Applied Sciences
University of Sri Jayewardenepura, Sri Lanka.
Email: ttalagala@sjp.ac.lk
Corresponding author

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: rob.hyndman@monash.edu

George Athanasopoulos
Department of Econometrics and Business Statistics,
Monash University, VIC 3145, Australia.
Email: george.athanasopoulos@monash.edu

23 March 2021

JEL classification: C10,C14,C22

Meta-learning how to forecast time series

Abstract

Features of time series are useful in identifying suitable models for forecasting. We present a general framework, labelled FFORMS (Feature-based FORecast Model Selection), which selects forecast models based on features calculated from the time series. The FFORMS framework builds a mapping that relates the features of a time series to the "best" forecast model using a random forest. The framework is evaluated using time series from the M-forecasting competitions and is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automated forecasting approaches. Using model-agnostic machine learning interpretability approaches we explore what is happening under the hood of the FFORMS framework. The analysis provides a more refined picture of the relationship between features and the choice of forecast model which is particularly valuable for ongoing research in the field of feature-based time series analysis.

Keywords: Algorithm selection problem, Black-box models, Random forest, Machine learning interpretability, Visualization

1 Introduction

Forecasting is a key activity for any business to operate efficiently. The rapid advances in computing technologies have enabled businesses to keep track of large numbers of time series. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for their thousands of products across various locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries, revenues, number of users for different services, etc. Such large collections of time series require fast automated procedures generating accurate forecasts. The scale of these tasks has raised some computational challenges we seek to address by proposing a new fast algorithm for model selection and time series forecasting.

Two alternative strategies for generating such a large number of forecasts are: (1) to either use a single forecasting method across all the time series; or (2) to select an appropriate forecasting method for each time series individually. It is highly unlikely that a single method will consistently outperform judiciously chosen competitor methods across all time series. We therefore reject the former strategy and focus on an approach for selecting an individual forecasting method for each time series.

Selecting the most appropriate model for forecasting a given time series can be challenging. Two of the most commonly used automated algorithms are the exponential smoothing (`ets`) algorithm of Hyndman et al. (2002) and the ARIMA (`auto.arima`) algorithm of Hyndman & Khandakar (2008). Both algorithms are implemented in the forecast package in R (R Core Team 2018; Hyndman et al. 2018). In this paradigm, a class of models is selected in advance, and many models within that class are estimated for each time series. The model with the smallest AICc value is chosen and used for forecasting. This approach relies on the expert judgement of the forecaster in first selecting the most appropriate class of models to use, as it is not usually possible to compare AICc values between model classes due to differences in the way the likelihood is computed, and the way initial conditions are handled.

An alternative approach, which avoids selecting a class of models *a priori*, is to use a simple "hold-out" test set; but then there is often insufficient data to draw a reliable conclusion. To overcome this drawback, time series cross-validation can be used (Racine 2000; Hyndman & Athanasopoulos 2018); then models from many different classes may be applied, and the model with the lowest cross-validated MSE selected. However, this increases the computation time involved considerably (at least to order n^2 where n is the number of series to be forecast).

Clearly, there is a need for a fast and scalable algorithm to automate the process of selecting models with the aim of forecasting. We refer to this process as forecast-model selection. We propose a general meta-learning framework using features of the time series to select the class of models, or even the specific model, to be used for forecasting. The forecast-model selection process is carried out using a classification algorithm — we use the time series features as inputs, and the "best" forecasting model as the output. The classifier is built using a large historical collection of time series, in advance of the forecasting task at hand. Hence, this is an "offline" procedure.

The "online" process of generating forecasts only involves calculating the features of a time series and using the pre-trained classifier to identify the best forecasting model. Hence, generating forecasts only involves the estimation of a single forecasting model, with no need to estimate

large numbers of models within a class, or to carry out a computationally-intensive cross-validation procedure. We refer to this general framework as FFORMS (**F**eature-based **FOR**ecast-**M**odel **S**election).

There have been several recent studies on the use of meta-learning approaches to automate forecast model selection based on features computed from the time series (Shah 1997; Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016). However, to the best of our knowledge, a very limited effort has been made to understand how the meta-learners are making their decisions and what is really happening inside these complex model structures. To fill these gaps, this paper makes a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques. We try to answer the following questions: i) **What** are the similarities and dissimilarities identified by the meta-learner between class-labels? ii) **Which** features contribute most to the classification processs? iii) **How** are features related to the property being modelled? iv) **How** do features interact with each other to identify a suitable forecasting model? We believe addressing these questions can enhance the transparency of the model predictions and build trust trust in model's predictions.

The remainder of the paper is structured as follows. We review the related work in Section 2. In Section 3 we explain the detailed components and procedures of our proposed framework for forecast model selection. Section 4 provides a description of machine learning interpretability techniques. In Section 5 we present the results, followed by the conclusions and future work in Section 6.

2 Related work

2.1 Time series features

Reid (1972) points out that the performance of forecasting methods changes according to the nature of the data. Exploring the reasons for these variations may be useful in selecting the most appropriate model. In response to the results of the M3 competition (Makridakis & Hibon 2000), similar ideas have been put forward by others. Hyndman (2001), Lawrence (2001) and Armstrong (2001) argue that the characteristics of a time series may provide useful insights into which methods are most appropriate for forecasting.

Rather than work with the time series directly at the level of individual observations, we propose

analysing time series via an associated "instance space given by the features". A time series feature is any measurable characteristic of a time series. For example, Figure 1 (left panel) shows the time-domain representation of six time series taken from the M3 competition (Makridakis & Hibon 2000) while Figure 1 (right panel) shows a feature-based representation of the same time series. Here only two features are considered: the strength of seasonality and the strength of trend, calculated based on the measures introduced by Wang, Smith-Miles & Hyndman (2009). Time series in the lower right quadrant of the scatter plot are non-seasonal but trended, while there is only one series with both high trend and high seasonality. We also see how the degree of seasonality and trend varies between series. Other examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and nonlinearity. Fulcher & Jones (2014) identified 9000 operations to extract features from time series.

The choice of the most appropriate set of features depends on both the nature of the time series being analysed, and the purpose of the analysis. In Section 5, we study the time series from the M1, M3 and M4 competitions (Makridakis et al. 1982; Makridakis & Hibon 2000; Makridakis, Spiliotis & Assimakopoulos 2019), and we select features for the purpose of forecast model selection. The M1, M3 and M4 competitions involve time series of differing length, scale and other properties. We include length as one of our features, but the remaining features are independent of scale and asymptotically independent of the length of the time series (i.e., they are ergodic). As our main focus is forecasting, we select features which have discriminatory power in selecting a good model for forecasting.

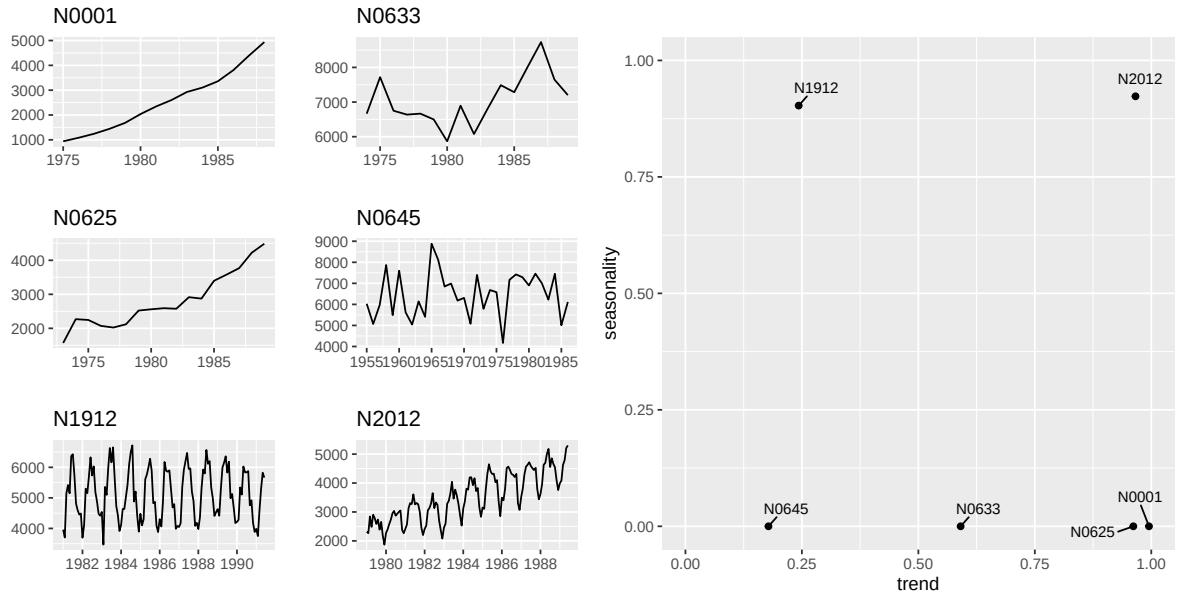

Figure 1: *Time-domain representation of time series (left) and feature-based representation of time series (right).*

2.2 Meta-learning for algorithm selection

John Rice was an early and strong proponent of the idea of meta-learning, which he called the algorithm selection problem (ASP) (Rice 1976). The term *meta-learning* started to appear with the emergence of the machine-learning literature.

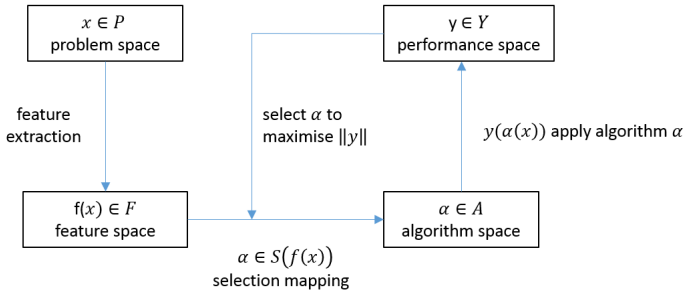

Figure 2: *Rice's framework for the Algorithm Selection Problem.*

Rice's framework for algorithm selection is shown in Figure 2 and comprises four main components. The problem space, P, represents the data sets used in the study. The feature space, F, is the range of measures that characterize the problem space P. The algorithm space, A, is a list of suitable candidate algorithms which can be used to find solutions to the problems in P. The performance metric, Y, is a measure of algorithm performance such as accuracy, speed, etc. The main challenge in ASP to identify the selection mapping S from the feature space to the algorithm space A. A formal definition of the algorithm selection problem is given by Smith-Miles (2009), and repeated below.

> **Algorithm selection problem**. For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

2.3 Forecast-model selection using meta-learning

Selecting models for forecasting can be framed according to Rice's ASP framework.

> **Forecast-model selection problem**. For a given time series $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into the algorithm space A, such that the selected algorithm $\alpha \in A$ minimizes forecast accuracy error metric $y(\alpha(x)) \in Y$ on the test set of the time series.

Existing methods differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

Collopy & Armstrong (1992) introduced 99 rules based on 18 features of time series, in order to make forecasts for economic and demographic time series. This work was extended by Armstrong (2001) to reduce human intervention.

Shah (1997) used the following features to classify time series: the number of observations, the ratio of the number of turning points to the length of the series, the ratio of number of step changes, skewness, kurtosis, the coefficient of variation, autocorrelations at lags 1–4, and partial autocorrelations at lag 2–4. Casting Shah's work in Rice's framework, we can specify: $P = 203$ quarterly series from the M1 competition (Makridakis et al. 1982); $A = 3$ forecasting methods, namely simple exponential smoothing, Holt-Winters exponential smoothing with multiplicative seasonality, and a basic structural time series model; $Y =$ mean squared error for a hold-out sample. Shah (1997) learned the mapping S using discriminant analysis.

Prudêncio & Ludermir (2004) was the first paper to use the term "meta-learning" in the context of time series model selection. They studied the applicability of meta-learning approaches for forecast model selection based on two case studies. Again using the notation above, we can describe their first case study with: A contained only two forecasting methods, simple exponential smoothing and a time-delay neural network; $Y =$ mean absolute error; F consisted of 14 features, namely length, autocorrelation coefficients, coefficient of variation, skewness, kurtosis, and a test of turning points to measure the randomness of the time series; S was learned using the C4.5 decision tree algorithm. For their second study, the algorithm space included a random walk, Holt's linear exponential smoothing and AR models; the problem space P contained the yearly series from the M3 competition (Makridakis & Hibon 2000); F included a subset of features from the first study; and Y was a ranking based on error. Beyond the task of forecast-model selection, they used the NOEMON approach to rank the algorithms (Kalousis & Theoharis 1999).

Lemke & Gabrys (2010) studied the applicability of different meta-learning approaches for time series forecasting. Their algorithm space A contained ARIMA models, exponential smoothing models and a neural network model. In addition to statistical measures such as the standard deviation of the de-trended series, skewness, kurtosis, length, strength of trend, Durbin-Watson statistics of regression residuals, the number of turning points, step changes, a predictability measure, nonlinearity, the largest Lyapunov exponent, and auto-correlation and partial-autocorrelation coefficients, they also used frequency domain based features. The feed forward

neural network, decision tree and support vector machine approaches were considered to learn the mapping S.

Wang, Smith-Miles & Hyndman (2009) used a meta-learning framework to provide recommendations as to which forecast method to use to generate forecasts. In order to evaluate forecast accuracy, they introduced a new measure $Y =$ *simple percentage better (SPB)*, which calculates the forecasting accuracy of a method against the forecasting accuracy error of random walk model. They used a feature set F comprising nine features: strength of trend, strength of seasonality, serial correlation, nonlinearity, skewness, kurtosis, self-similarity, chaos and periodicity. The algorithm space A included eight forecast models, namely, exponential smoothing, ARIMA, neural networks and random walk model; while the mapping S was learned using the C4.5 algorithm for building decision trees. In addition, they used SOM clustering on the features of the time series in order to understand the nature of time series in a two-dimensional setting.

The set of features introduced by Wang, Smith-Miles & Hyndman (2009) was later used by Widodo & Budi (2013) to develop a meta-learning framework for forecast-model selection. The authors further reduced the dimensionality of time series by performing principal component analysis on the features.

More recently, Kück, Crone & Freitag (2016) proposed a meta-learning framework based on neural networks for forecast-model selection. Here, $P = 78$ time series from the NN3 competition were used to build the meta-learner. They introduced a new set of features based on forecasting errors. The average symmetric mean absolute percentage error was used to identify the best forecast models for each series. They classify their forecast models in the algorithm space A, comprising single, seasonal, seasonal-trend and trend exponential smoothing. The mapping S was learned using a feed-forward neural network. Further, they evaluated the performance of different sets of features for forecast-model selection.

3 Methodology

Our proposed FFORMS framework, presented in Figure 3, builds on this preceding research. The offline and online phases are shown in blue and red respectively. A classification algorithm (the meta-learner) is trained during the offline phase and is then used to select an appropriate forecast model for a new time series in the online phase. We use machine learning interpretability tools to gain insights into what is happening under the hood of the FFORMS framework. We refer to this process as "peeking inside FFORMS".

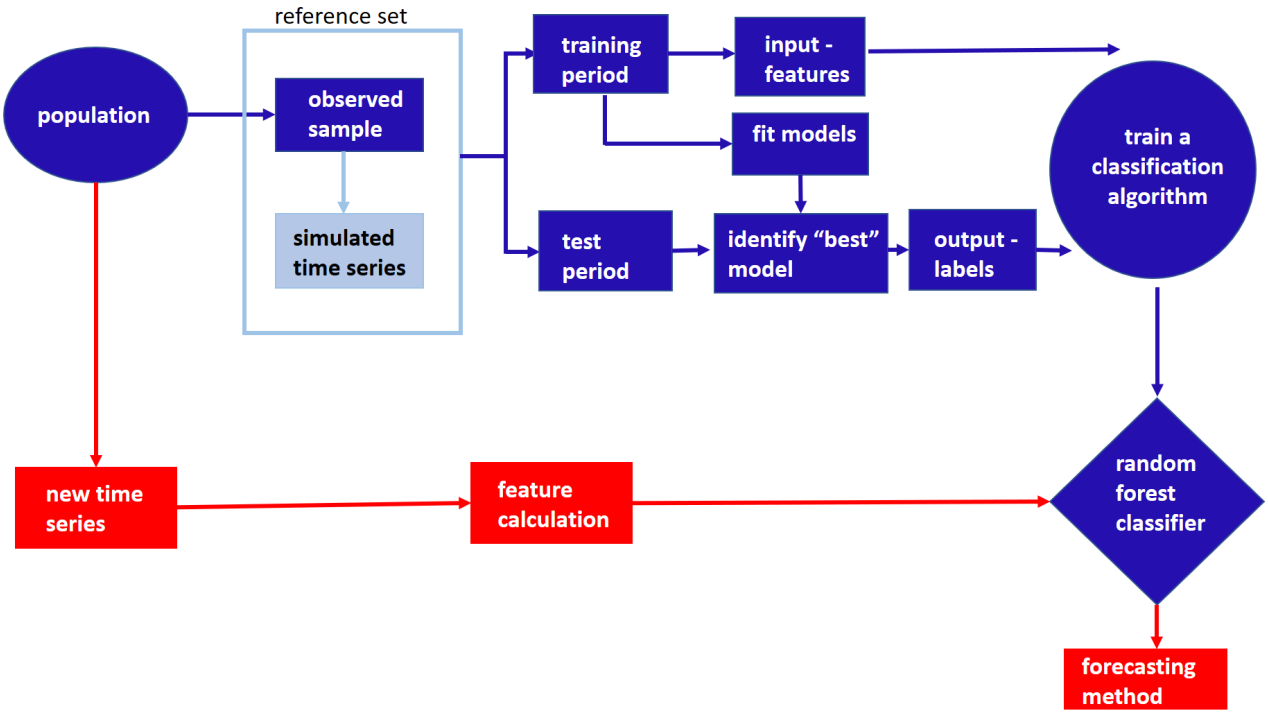

Figure 3: *FFORMS (Feature-based FORecast-Model Selection) framework. The offline phase is shown in blue, the online phase is in red.*

In order to train our classification algorithm, we need a large collection of time series which are similar to those we will be forecasting. We assume that we have an essentially infinite population of time series, and we take a sample of them in order to train the classification algorithm denoted as the "observed sample". The new time series we wish to forecast can be thought of as additional draws from the same population. Hence, any conclusions made from the classification framework refer only to the population from which the sample has been selected. We may call this the "target population" of time series. It is important to have a well-defined target population to avoid misapplying the classification rules. In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population (we provide details and discussion in Section 3.1 that follows). We denote the total collection of time series used for training the classifier as the "reference set". Each time series within the reference set is split into a training period and a test period. From each training period we compute a range of time series features, and fit a selection of candidate models. The calculated features form the input vector to the classification algorithm. Using the fitted models, we generate forecasts and identify the "best" model for each time series based on a forecast error measure (e.g., MASE) calculated over the test period. The models deemed "best" form the output labels for the classification algorithm. The pseudo code for our proposed framework is presented in Algorithm 1 below. In the following sections, we briefly discuss

aspects of the offline phase.

Algorithm 1 The FFORMS framework - Forecasting based on meta-learning.

Offline phase - train the classifier

Given:

$O = \{x_1, x_2, \ldots, x_n\}$: the collection of n observed time series.
L : the set of class labels (e.g. ARIMA, ETS, SNAIVE).
F : the set of functions to calculate time series features.
$nsim$: number of series to be simulated.
B : number of trees in the random forest.
k : number of features to be selected at each node.

Output:

FFORMS classifier

Prepare the reference set

For $i = 1$ to n:

1: Fit ARIMA/ETS (for yearly, quarterly, monthly), or MSTL(for weekly, daily, hourly) models to x_i.
2: Simulate $nsim$ time series from each model in step 1.
3: The time series in O and simulated time series in step 2 form the reference set $R = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_N\}$ where $N = n + nsim$.

Prepare the meta-data

For $j = 1$ to N:

4: Split x_j into a training period and test period.
5: Calculate features F based on the training period.
6: Fit L models to the training period.
7: Calculate forecasts for the test period from each model.
8: Calculate forecast error measure over the test period for all models in L.
9: Select the model with the minimum forecast error.
10: Meta-data: input features (step 5), output labels (step 9).

Train a random forest classifier

11: Train a random forest classifier based on the meta-data.
12: Random forest: the ensemble of trees $\{T_b\}_1^B$.

Online phase - forecast a new time series

Given:

FFORMS classifier from step 12 .

Output:

class labels for new time series x_{new}.

13: For x_{new} calculate features F.
14: Let $\hat{C}_b(x_{new})$ be the class prediction of the b^{th} random forest tree. Then class label for x_{new} is $\hat{C_{rf}}(x_{new}) = majorityvote\{\hat{C}_b(x_{new})\}_1^B$.

3.1 Augmenting the observed sample with simulated time series

In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population. This process may be useful when the number of observed time series is too small to build a reliable classifier. Alternatively, we may wish to add more of some particular types of time series to the reference set in order to get a more balanced sample for the classification. In order to produce simulated series that are similar to those in the population, we consider three classes of data generating processes: exponential smoothing models, ARIMA models and multiple seasonal decomposition approach. The `ets` and `auto.arima` functions available in the forecast package in R (Hyndman et al. 2018) are

used to simulate yearly, quarterly and monthly data from exponential smoothing and ARIMA models. The `stlf` function, also available in the forecast package is used to simulate time series based on multiple seasonal decomposition (MSTL) approach for weekly, daily and hourly series. Assuming the models produce data that are similar to the observed time series ensures that the simulated series are similar to those in the population. As this is done in the offline phase of the algorithm the computational time in producing these simulated series is of no real consequence.

3.2 Input: features

A comprehensive description of the features used in the experiment is presented in Table 1. A feature of the proposed FFORMS framework is that its online phase is fast compared to the time required for implementing a typical model selection or cross-validation procedure. Therefore, we consider only features that can be computed rapidly, as this computation must be done during the online phase. Furthermore, all our features are easily interpretable.

3.3 Output: labels

The task of the classification algorithm is to identify the "best" forecasting method for a given time series. The candidate models considered as labels will depend on the observed time series. For example, if we have only non-seasonal time series, and no chaotic features, we may wish to restrict the models to white noise, random walk, ARIMA and ETS processes. We fit the corresponding models outlined in Table 2 to each series in the reference set.

Each candidate model considered is estimated strictly on the training period of each series in the reference set. Forecasts are then generated for the test period over which the chosen forecast accuracy measure is calculated. The model with the lowest forecast error measure over the test period is deemed "best". This step is the most computationally intensive and time-consuming, as each candidate model has to be applied on each series in the reference set. However, as this task is done during the offline phase of the FFORMS framework, the time involved and computational cost associated are of no real significance and are completely controlled by the user.

According to the *M4 Competitor's Guide* (2018), in the M4 competition the forecast accuracy is evaluated based on the MASE and the symmetric mean absolute percentage error (sMAPE). Hence, in our applications, to identify the best forecast model for each time series, both forecast error measures, MASE and sMAPE, are calculated for each of the forecast models. Each of these is standardised by the median MASE and median sMAPE, calculated across the forecast models.

Table 1: *Time series features*

	Feature	Description	Y	Q/M	W	D/H
1	T	length of time series	✓	✓	✓	✓
2	trend	strength of trend	✓	✓	✓	✓
3	seasonality_q	strength of quarterly seasonality	-	✓	-	-
4	seasonality_m	strength of monthly seasonality	-	✓	-	-
5	seasonality_w	strength of weekly seasonality	-	-	✓	✓
6	seasonality_d	strength of daily seasonality	-	-	-	✓
7	seasonality_y	strength of yearly seasonality	-	-	-	✓
8	linearity	linearity	✓	✓	✓	✓
9	curvature	curvature	✓	✓	✓	✓
10	spikiness	spikiness	✓	✓	✓	✓
11	e_acf1	first ACF value of remainder series	✓	✓	✓	✓
12	stability	stability	✓	✓	✓	✓
13	lumpiness	lumpiness	✓	✓	✓	✓
14	entropy	spectral entropy	✓	✓	✓	✓
15	hurst	Hurst exponent	✓	✓	✓	✓
16	nonlinearity	nonlinearity	✓	✓	✓	✓
17	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓	✓	-
18	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓	✓	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓	-	-
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓	-	-
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓	-	-
22	ur_pp	test statistic based on Phillips-Perron test	✓	-	-	-
23	ur_kpss	test statistic based on KPSS test	✓	-	-	-
24	y_acf1	first ACF value of the original series	✓	✓	✓	✓
25	diff1y_acf1	first ACF value of the differenced series	✓	✓	✓	✓
26	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓	✓	✓
27	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓	✓	✓
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓	✓	✓
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓	✓	✓
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-	✓	✓	✓
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓	✓	✓
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓	✓	✓
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓	✓	✓
34	lmres_acf1	first ACF value of residual series of linear trend model	✓	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓	✓	✓
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓	✓	✓
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓	✓	✓

The model with the lowest average value of the scaled MASE and scaled sMAPE is selected as the output class label.

3.4 Train a FFORMS meta-learner

A random forest algorithm is used to train the meta-learner. The motivation behind the use of random forest classifiers are: i) it can model complex interactions between features; ii) the modelling framework captures linear and non-linear effects of features through the averaging of large number of decision trees; iii) its robustness against over-fitting the training data; iv) it is easy to handle the problem of imbalanced classes; v) it is a fast approach compared to boosting algorithms and vi) it is fairly easy and straightforward to implement with available software. In

Table 2: *Class labels*

Class label	Description	Y	Q/M	W	D/H
wn	white noise process	✓	✓	✓	✓
ARMA	AR, MA, ARMA processes	✓	✓	✓	-
ARIMA	ARIMA process	✓	✓	✓	-
SARIMA	seasonal ARIMA	-	✓	✓	-
rwd	random walk with drift	✓	✓	✓	✓
rw	random walk	✓	✓	✓	✓
theta	standard theta method	✓	✓	✓	✓
stlar	seasonal decomposition with AR modelling of the seasonally adjusted series	-	✓	✓	✓
ETS_NTNS	ETS without trend and seasonal components	✓	✓	-	-
ETS_T	ETS with trend component and without seasonal component	✓	✓	-	-
ETS_DT	ETS with damped trend component and without seasonal component	✓	✓	-	-
ETS_TS	ETS with trend and seasonal component	-	✓	-	-
ETS_DTS	ETS with damped trend and seasonal component	-	✓	-	-
ETS_S	ETS with seasonal component and without trend component	-	✓	-	-
snaive	seasonal naive method	-	✓	✓	✓
tbats	TBATS forecasting	-	✓	✓	✓
nn	neural network time series forecasts	✓	✓	✓	✓
mstlets	multiple seasonal decomposition with ETS modelling of the seasonally adjusted series	-	-	✓	✓
mstlarima	multiple seasonal decomposition with ARIMA modelling of the seasonally adjusted series	-	-	-	✓

this work, we have used the randomForest package (Liaw & Wiener 2002; Breiman et al. 2018) which implements the Fortran code for random forest classification by Breiman & Cutler (2004). To determine the class label for a new instance, features are calculated and passed down the trees. Then each tree gives a prediction and the majority vote over all individual trees leads to the final decision.

The random forest (RF) algorithm is highly sensitive to class imbalance (Breiman 2001), and our reference set is unbalanced: some classes contain significantly more cases than other classes. The degree of class imbalance is reduced to some extent by augmenting the observed sample with the simulated time series. We consider three approaches to address the class imbalance in the data: (1) incorporating class priors into the RF classifier; (2) using the balanced RF algorithm introduced by Chen, Liaw & Breiman (2004); and (3) re-balancing the reference set with down-sampling. In down-sampling, the size of the reference set is reduced by down-sampling the larger classes so that they match the smallest class in size; this potentially discards some useful information. Comparing the results, the balanced RF algorithm and RF with down-sampling did not yield satisfactory results. We therefore only report the results obtained by the RF built on unbalanced data (RF-unbalanced) and the RF with class priors (RF-class priors). The RF algorithms are implemented by the randomForest R package (Liaw & Wiener 2002; Breiman et al. 2018). The class priors are introduced through the option `classwt`. We use the reciprocal of class size as the class priors. The number of trees `ntree` is set to 1000, and the number of randomly selected features `k` is set to be one third of the total number of features available.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class, but in finding the best possible forecast model. It is possible that two models produce almost equally accurate forecasts, and therefore it is not important whether the classifier picks one model over the other. Therefore we report the forecast accuracy obtained from the FFORMS framework, rather than the classification accuracy.

4 Peeking inside FFORMS

4.1 Application to M4-competition data

To test how well our proposed framework can identify suitable models for forecasting, we use the time series from the M1 (Makridakis et al. 1982), M3 (Makridakis & Hibon 2000) and M4 (Makridakis, Spiliotis & Assimakopoulos 2019) competitions. The M-competition data were a sample of time series collected from several domains such as demographics, finance, business and economics. In our experiment, we treat the time series from the M1 and M3 competitions as the observed sample. We augment the reference set by adding multiple time series simulated using the training period of each series in the M4 competition. We build separate FFORMS meta-learners to yearly, quarterly, monthly, daily and hourly series due to their differences in features and classlabels. Then the pre-trained FFROMS meta-learners are used to forecast the test period of the M4 competition time series.

5 Summary of the main results

Table 3 shows the performance of the FFORMS approach on the M4 competition data. The point forecasts and prediction intervals are evaluated based on the test period of each series. The point forecasts are evaluated based on the MASE computed for each forecast horizon, and then by averaging the MASE errors across all series corresponding to each frequency category. Similarly, MSIS is used to evaluate the prediction intervals. The results are compared against several benchmarks and the top three places of the M4 competition. The top-ranking methods of the M4 competition are based on some kind of combination approach. The first ranking method is based on a hybrid approach that produces forecasts based on both ETS and machine learning approaches. The second and third places are based on a combination of nine and seven different forecast models. The second and third approaches are time-consuming because forecasts from all candidate models must be computed. The results of the FFORMS approach are based on

Table 3: *The performance of FFORMS on the M4 competition data based on point forecasts (based on MASE) and prediction intervals (based on MSIS)*

Point Forecasts (Mean Absolute Scaled Error (MASE))						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.17	1.20	0.98	2.31	3.57	0.84
FFORMS-com	3.23	1.16	0.94	2.55	3.42	0.81
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	3.07	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	3.27	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	3.27	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4-1st	**2.98**	1.12	**0.88**	2.36	3.45	0.89
M4-2nd	3.06	**1.11**	0.89	**2.11**	3.34	**0.81**
M4-3rd	3.13	1.12	0.91	2.16	**2.64**	0.87
Prediction Intervals (Mean Scaled Interval Score (MSIS))						
FFORMS	39.79	11.24	9.82	**20.84**	36.36	8.0
M4-1st	**23.89**	**8.55**	**7.20**	22.03	**26.28**	**7.92**
M4-2nd	27.47	9.38	8.65	21.53	34.37	18.50
M4-3rd	not submitted					
naive	56.55	14.07	12.30	26.35	32.55	71.24

Note: Bold signifies the best performing method.

individual forecasts. According to the Table 3, we can see the FFORMS approach achieved comparable performance in a much more cost and time-effective way. Based on the FFORMS approach, we provided an entry to the M4 competition. After submission we found a bug in our implementation of hourly data, hence only the hourly data results are different from the published results of the M4 competition. The yearly, quarterly, monthly, weekly and daily result are same as the published results. This completes the evaluation of the FFORMS framework.

The main question we have now is: 'Can we trust the FFORMS framework if we do not know how it works?' In the next sections we present results showing **"what is happening under the hood of FFORMS?"**. Note that, the results for quarterly and weekly data are very similar to the corresponding results of monthly data and the results for daily data are very similar to the corresponding plots of hourly data. Hence, the results correspond to quarterly, weekly and daily data are not presented in the subsequent analysis. We present the results for yearly series; represents non-seasonal time series, monthly series: represents time series with single seasonal component and hourly series: represents series with multiple seasonal components.

5.1 Visualization of FFORMS's understanding of the similarity and dissimilarity between forecast models

In FFORMS, the forecast-model selection problem is posed as a supervised learning task. Each time series in the meta-data set is represented as a vector of features and labelled according to the "best" forecast model. Then the FFORMS is used to identify the "best" forecast model given a vector of features of the time series. For example, suppose for a given time series the forecast error vector for the random walk, random walk with drift and white noise process is [1.31, 1.30, 3.40]. Then, ideally the vote probability vector we expect from FFORMS is [0, 1, 0], .i.e, the best model is given class probability 1 and others 0. However, due to similarities in class-labels and limitations in features we get non-zero vote probabilities for other class-labels as well. This is not a big issue as long as the random forest assign high probability to the forecast models that are very similar to best forecast model and very low probability to models that are very different to the best forecast model. For example, in the afformentioned situation, if relatively high probability is assigned to both random walk with drift and random walk and very low probability for white noise, then it indicates FFORMS algorithm correctly picked up the similarities and dissimilarities between class labels and our features are useful in revealing those similarities and dissimilarities. Hence, by exploring vote-matrix probabilities for all time series we can identify the FFORMS's understanding of the similarities and dissimilarities between forecast models.

We generated several initial plots to visualise the vote matrix calculations based on OOB observations to get a better understanding of the relationships learned by the random forest. Out of different plots we created "heatmap" seems to be the most useful in visualising vote matrix probabilities. We use `iheatmapr` function in `iheatmapr` package in R to visualise interactive heatmap (Schep & Kummerfeld 2017). Furthermore, to improve the visualization we reorder and cluster the rows and columns of the matrix separately by applying hierarchical clustering approach. The results are shown in Figure 4, Figure 5 and Figure 6. Visualising the vote matrix probabilities in this manner helps us to identify several important characteristics learned by the random forest algorithm.

According to Figure 4, Figure 5 and Figure 6, in general we can see the columns are clustered accroding to the similarities of forecast models. The results are summarized in Table 4. An interpretation for each cluster is given in blod text. Observing the yearly results we can see class labels are classified as according to their capability in handling trend, for example white noise process, ARMA, ETS models with damped trend into one cluster and ARIMA and ETS with

trend into a separate cluster. Furthermore, for monthly series class labels are correctly classified according to the similarities in trend and seasonal component. In hourly series, vote matrix probabilities shows similarities between models that can handle multiple seasonal component. Hence, this indicates our meta-learner correctly identified the similarities and dissimilarities between classes.

Table 4: *Summary of column clusters displayed in vote matrices of Figure 4- 6*

Yearly	Monthly	Hourly
Column Cluster 1	*Column Cluster 1*	*Column Cluster 1*
Models flexible for any situation	**Models flexible for any situation**	**Models without seasonal component and Theta method**
Neural network	SARIMA	Random walk
Random walk	tbats	Random walk with drift
Theta	Theta	Theta
		White noise process
Column Cluster 2	*Column Cluster 2*	*Column Cluster 2*
Models not suitable to handle highly trended series	**Models that can handle series with complex patterns**	**Models that can handle multiple seasonality**
ARMA/AR/MA	Neural networks	matlarima
ETS with damped trend	Random walk with drift	mstlets
ETS without trend and seasonality	STLAR	
White noise process		
Column Cluster 3	*Column Cluster 3*	*Column Cluster 3*
Random walk with drift	**Models without seasonal component**	**Models for time varying seasonality**
Random walk with drift	ARIMA	snaive
	ARMA/AR/MA	STLAR
	ETS-damped trend	TBATS
	ETS without trend and seasonality	
	ETS with trend	
	Random walk	
	White noise process	
Column Cluster 4	*Column Cluster 4*	*Column Cluster 4*
Models that can handle highly trended series	**ETS models with seasonal component and snaive**	**Neural network**
ARIMA	ETS with damped trend and seasonality	Neural network
ETS with trend	ETS with seasonal component	
	ETS with trend and seasonality	
	snaive	

- Irrespective of the true classlable, the random walk with drift has been assigned a non-zero vote probability for all yearly series. This means, for all of the yearly series at least one tree in the random forest voted for random walk with drift. According to the results in Table 3 also we can see random walk with drift perform extremely well across all time series.
- For monthly series in general, SARIMA, tbats, theta, nn, rwd and stlr have been assigned a very high vote probability.
- Zooming out the results reveal that the monthly series that have assigned high vote probability to SARIMA have also assigned a high vote probability to ETS models with

seasonal components. Furthermore, the series that have assigned high vote probability to `ARIMA` models have also assigned a high probability to `ETS-dampedtrend` and `ETS-trend`.

- Compared to yearly and monthly series, in hour series classification the best forecasting model has been selected with very high probability.
- In hourly series very low vote proabilities have been assigned to rw, rwd, theta and wn and most of the series nn has been assigned a non-zero vote probability.

We also created an interactive dashboard by adding an interactive layer to the vote matrix and visualizing other related plots of the vote matrix clusters such as: i) distribution of true classlabels by row clusters of the vote matrix, ii) feature distribution by clusters and iii) distribution of the clusters in the feature space (principal component analysis is used to map each time series as a point in a two-dimensional instance space given by features). The interactive dashboard showcasing our results is available online at https://thiyangt.github.io/fformsviz/fforms.html. Interactive dashboard visualisation reveals that there is a matching correspondence between the vote probability matrix and the composition of clusters by true class labels. For example, in cluster 1 and cluster 2 the majority of time series are labelled as random walk with drift (rwd), ARIMA and ETS with trend (ETS-trend). In the vote probability matrix, high probability valuess have been assigned to those three classes. Furthermore, in cluster 3 the majority of time series are labelled as rwd, and the series in cluster 3 have selected random walk with drift with high vote probability. In cluster 4, the majority of series are in rwd and ETS-trend and they have been assigned a very high vote probability. In cluster 5, neural network (nn), white noise(wn) and rwd have been assigned a very high probability and the majority of time series in cluster 5 are classified as nn, wn or rwd. This indicates that our meta-learner correctly predicts the classlabels. Furthermore, the dashboard visualisation shows that the vote matrix visualisations of quarterly data depicted a similar pattern to monthly data. For quarterly and monthly data, the same set of features and class labels are used to train the model, Hence, this consistency between the results of the quarterly and the monthly series would provide evidence in support of the validity and trustability of the meta-learning framework.

5.2 Which features are the most important and where are they important?

Figure 7 shows top-5 most important features the within each class across yearly, monthly and daily series. The main point here is strength of trend, sum of first five partial autocorrelation coefficients, stability, the first ACF value of the differenced series and linearity are the most important features across many classes and frequency categories. The information about the

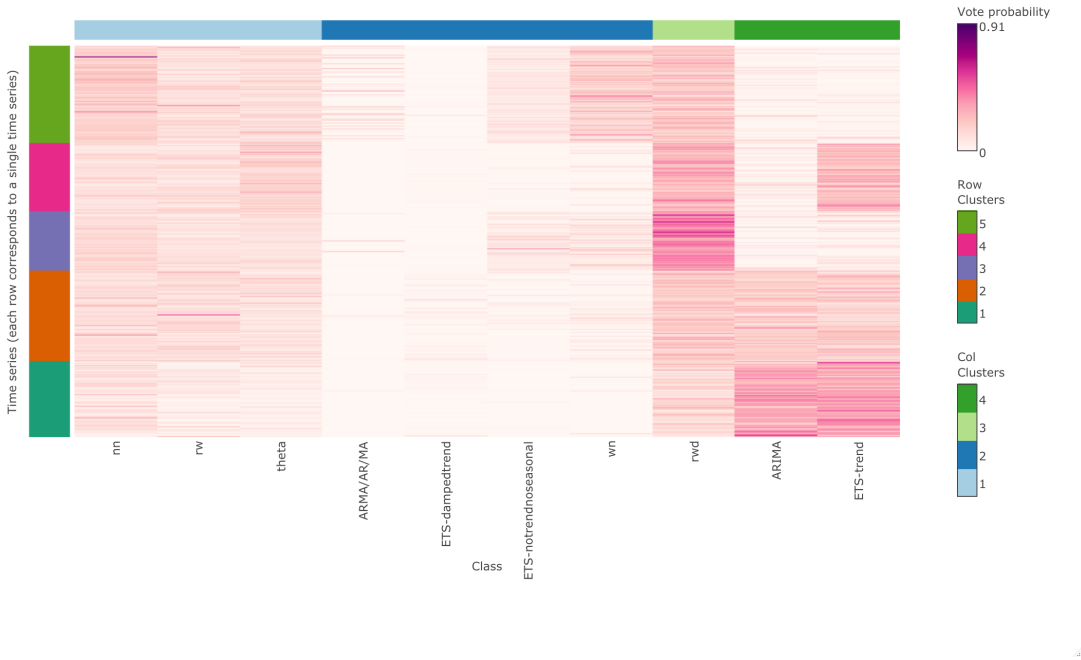

Figure 4: *Visualization of vote matrix of yearly series. The X-axis denotes the class labels.*

strength of trend in the series, linearity and sum of first five partial autocorrelation are very important when selecting white noise process among all frequency categories. For yearly data, the test statistic based on the Phillips-Perron unit root test has been ranked among top-5 within all classes. In addition to linearity, the other features related to different types of trend (damped trend, measured by beta, and exponential trend, measured by curvature) are assigned a very high importance within ETS_T, ETS_DT and ETS_NTNS, which handle different versions of trend within the ETS family. The length of time series (T) and stability are assigned a very high importance within monthly series. For seasonal time series strength of seasonality has been selected among top-5 features across all classes. For hourly data, the strength of daily seasonality (measured by seasonal_d) appears to be more important than the strength of weekly seasonality (measured by seasonal_w). In addition to the strength of seasonality, the other features such as ACF, PACF-based features related to seasonal lag and seasonally differenced series are also ranked among top five within some classes.

5.3 When and how are features linked to the prediction outcome?

For yearly series, except ETS_NTNS, Phillips-Perron test statistic is the most important feature across all classes. Figure 8 shows the partial dependency curves of the of the Phillips-Perron test statistic. Within each class the displayed relationship is consistent with the theoretical expectations. For example, a high negative value of the Phillips-Perron test statistic indicates

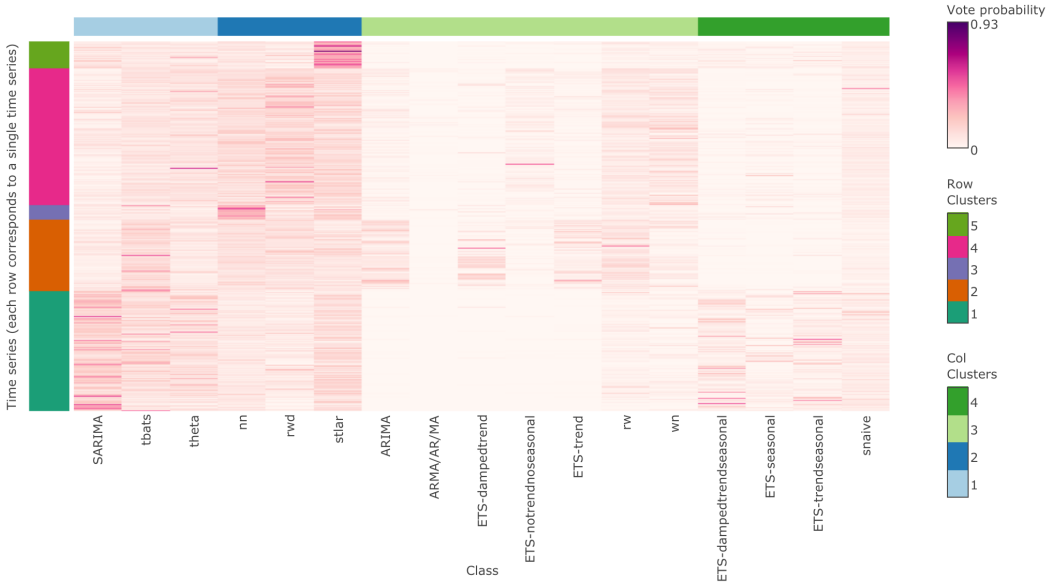

Figure 5: *Visualization of vote matrix of monthly series. The X-axis denotes the class labels.*

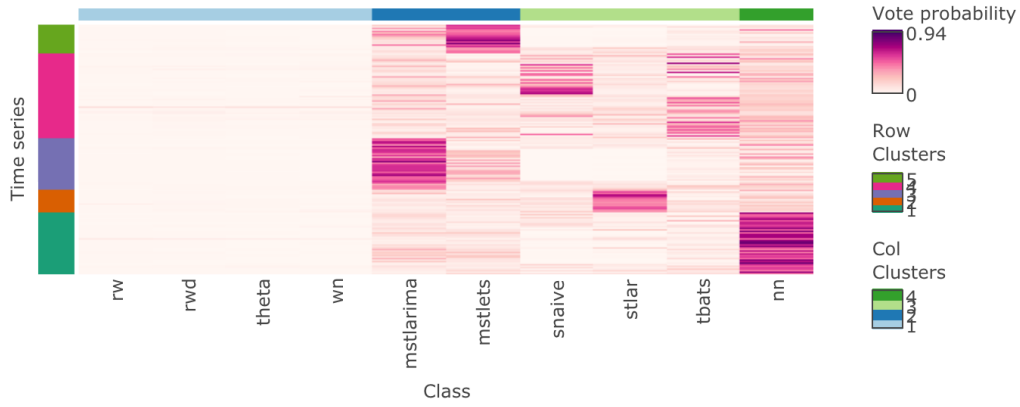

Figure 6: *Visualization of vote matrix of hourly series. The X-axis denotes the class labels.*

a stationarity of the series, while a positive value of the test statistic indicates nonstationarity of the series. According to Figure 8, we can see that the probability of selecting an ETS model with a trend component and ARIMA models increases steadily as ur_pp increases, while the opposite relationship could be observed for ARMA and WN.

For monthly series, the strength of seasonality is the most important features across all classes. According to Figure 9, it is immediately apparent that the probability of selecting a seasonal

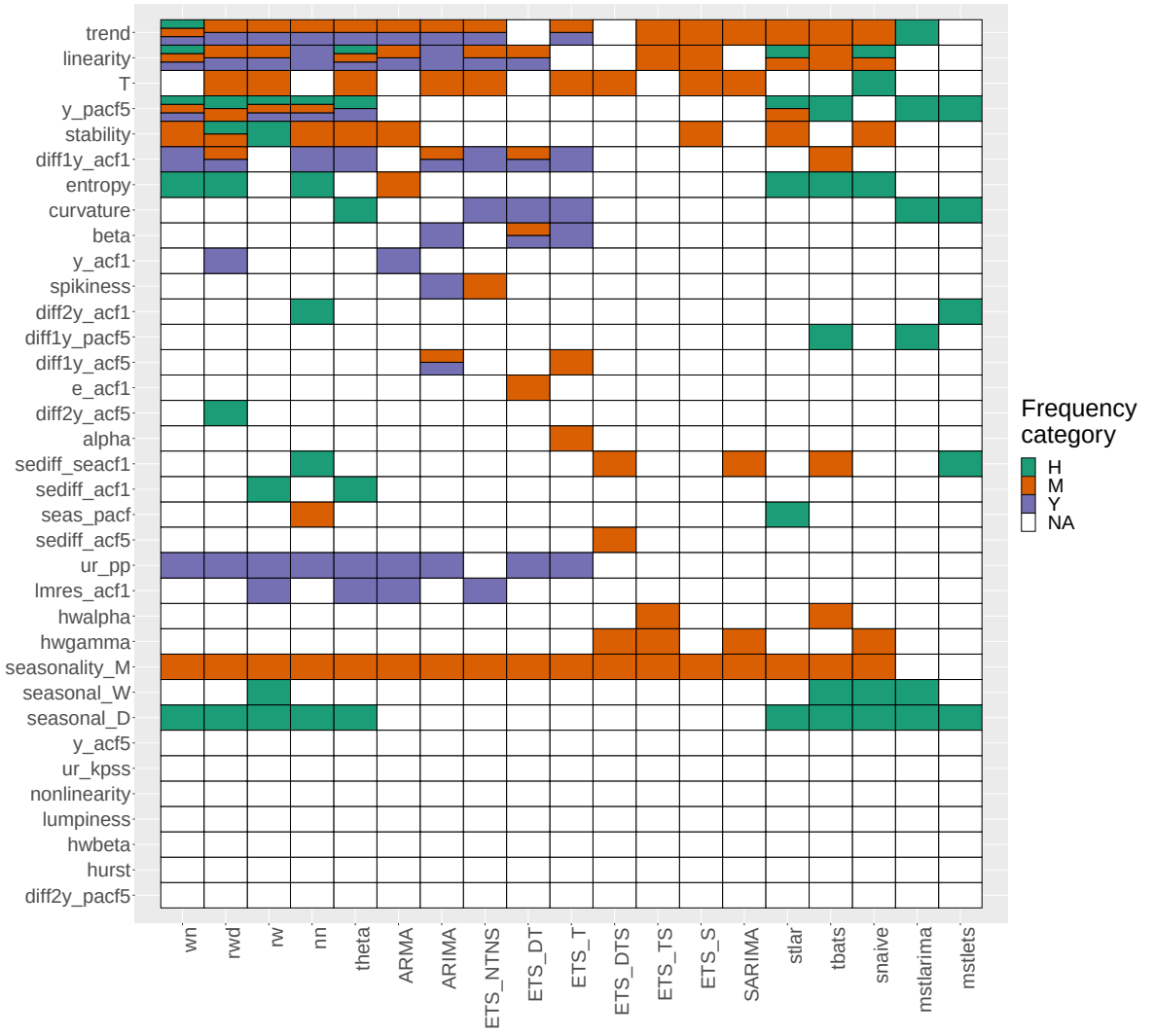

Figure 7: *Visualisation of top five features across frequency categories and classes. The X-axis denotes the features and Y-axis denotes the forecast model classes. The cell colours denote the frequency group in which the fetures ranked among top five. Strength of trend, linearity and strength of seasonality have been selected among top five features across many classes.*

forecast model increases as the strength of seasonality increases. The length of time series is also assigned a high importance within many classes. Partial dependency curves of Figure 10 reveal longer time series tend to select highly parameterized models or models involving seasonal components such as (ETS_TS, ETS_DTS, ETS_S, SARIMA, tbats), while shorter series tend to select simple models such as random walk, random walk with drift etc.

The partial dependency plots of the strength of seasonalities for hourly series are shown in Figure 11. According to Figure 11, the probability of selecting random walk, random walk with drift, theta and white noise process decreases with a higher value of strength of daily seasonality. On the other hand, the probability of the selecting random walk model increases as the strength of weekly seasonality increases. Figure 12 show the partial dependency plot

of stability. One notable difference between the yearly series and monthly series is that for monthly series, the feature stability is ranked among the top five within many classes. The feature stability does not appear to be ranked among the top 5 within any of the classes for yearly series. The stability is ranked important only within random walk and random walk with drift for hourly series. According to the Figure 12 we can see that the probability of selecting white noise models increases as stability increases for monthly series while for hourly series the probability of selecting random walk models and random walk with drift models increases as stability increases. The feature diff1y_acf1 is useful in determining the difference stationarity of the time series. This feature has been ranked among the top-five within the classes of yearly and monthly series. The associated partial dependency plots are shown in Figure 13. It is interesting to observe that difference stationary series are less likely to select ARIMA models. The reason could be difference stationary series are more likely to select random walk models, and ETS models. Furthermore, partial difference curve corresponds to white noise class shows difference stationary series have high chance of selecting white noise models. This unusual pattern could be due to over-differencing of the series or interaction effect between features.

The features, sum of the first five partial autocorrelation coefficients, strength of trend and linearity are the most commonly selected features within many classes across all three frequency category. The partial dependency plot for the sum of the first five partial autocorrelation coefficients are shown in Figure 14. All curves shows a turning point in the relations around 1. It also appears that the pattern of the relationship varies across classes as well as frequency categories. Figure 15 show the partial dependency curves for strength of trend. The probability of selecting an ETS model without trend components: ETS (N,N,N), white noise process and ARMA is extremely low when the strength of trend is extremely high, while the opposite relationship can be observed for other classes. The partial dependency plot for linearity is shown in Figure 16. According to Figure 16 for yearly series random walk with drift and ARMA classes are highly sensitive to the value of linearity around 0. However, the partial dependency curves of linearity for other combinations are relatively flat throughout the range. This could be due to the interaction effect of linearity with other features.

The two-way partial dependency plots of linearity are explored to see how linearity behave in the presence of other top-5 features. The associated two-way partial dependency plots are shown in Figure 17, Figure 18 and Figure 19 for yearly, monthly and hourly series respectively. According to the Figure 17 within wn, random walk, theta and nn classes linearity shows a pattern of interactivity with lmres_acf1 and diff1y_acf1. Within both ARIMA and ARMA classes main effect of linearity dominates. This is consistent with partial dependency curves we

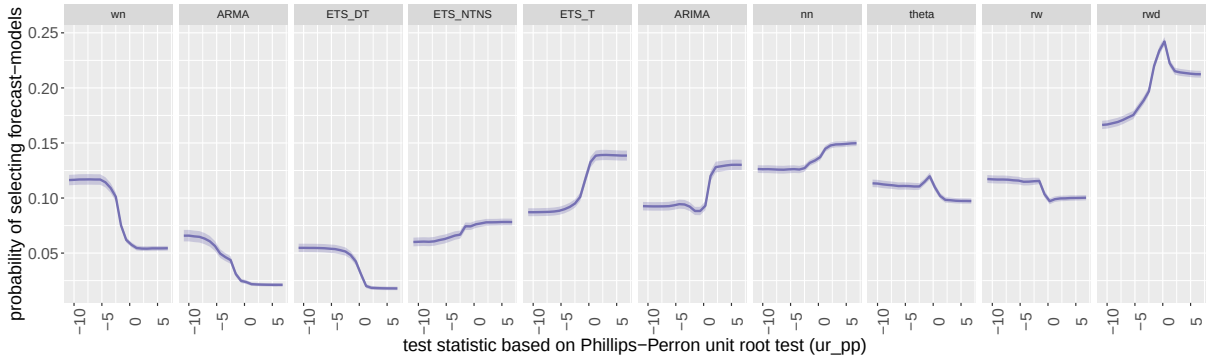

Figure 8: *Partial dependence plots for Phillips-Perrorn unit root test statistic. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. All classes show a turning point in the relationship around zero.*

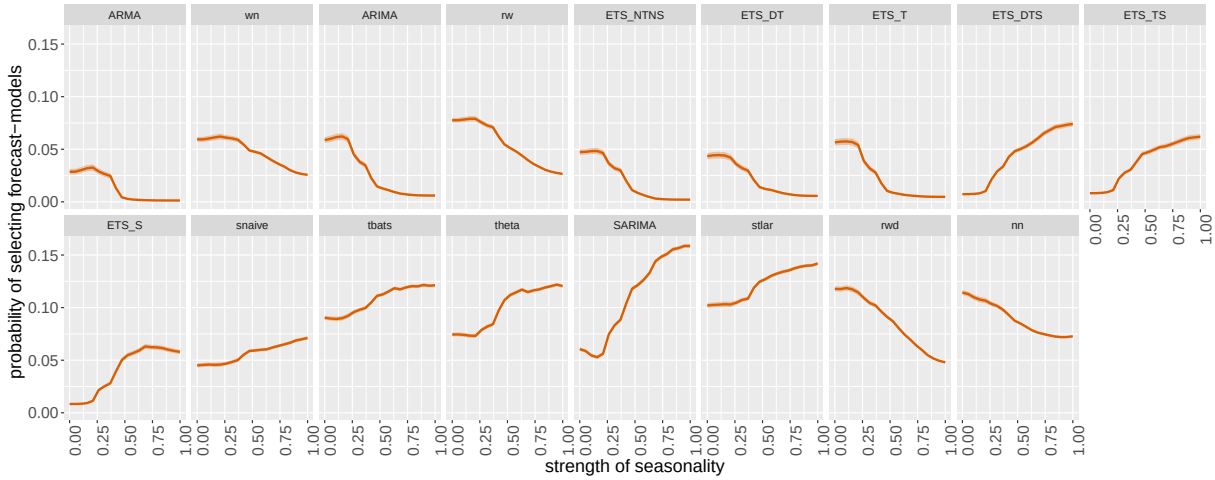

Figure 9: *Partial dependence plots for strength of seasonality for monthly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting forecast models with seasonal components increases as seasonality increases.*

observed in Figure 16. For monthly series, linearity shows interactivity with acf value at the first seasonal lag of seasonally-differenced series, stability, first ACF value at the remainder series and a parameter estimate of ETS(A, A, A) model. According to Figure 18, we can see there is a unique pattern of interactivity existing within each class. The two-way partial dependency plot for hourly series between the ACF value at the first seasonal lag of seasonally-differenced series and linearity are shown in Figure 19. The inconsistent level of colour intensity throughout the panels in tbats, nn and stlar indicate probability of selecting the corresponding forecast models changes according to the changes in both the features. Within rw and mstlarima we can see a separation between the lower half and the upper half due to the main effect of sediff_seacf1. The partial dependency curves of theta for stability is relatively flat. Figure 20 shows how trend, length, first autocorrelation coefficient of the differenced series interact with stability within different ranges. For example, Figure 20 as length of time series decreases, a pattern of interaction is visible with stability.

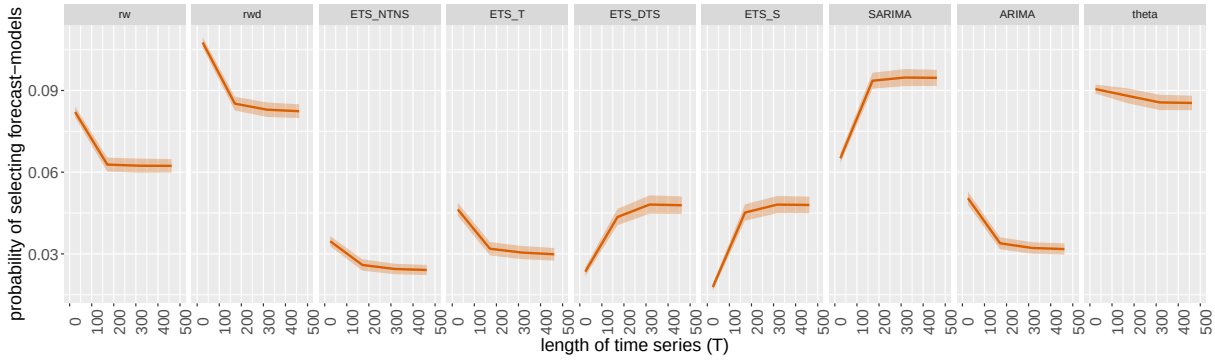

Figure 10: *Partial dependence plots for length of time series (T). The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting rw and rwd decreases as the length > 500.*

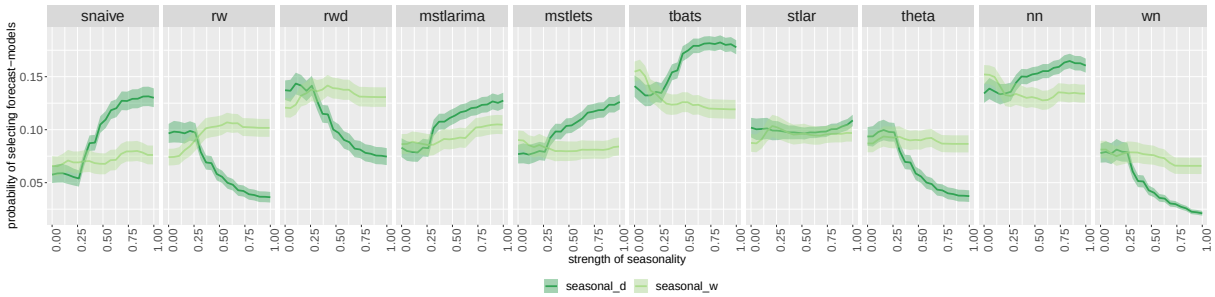

Figure 11: *Partial dependence plots for strength of seasonality for hourly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

FFORMS framework classified some series with very high probability (greater than 0.6), which indicates for a given series, majority of the trees voted for the same class. Hence, these instances can be considered as series that are easy to classify. We next try to visualize the distribution of these instances in the feature space. This is achieved by projecting the training dataset into a meaningful low-dimensional feature space and then visualize locations of the series that are classified with very high probabilities. This approach is a visualisation of model in the data space (Wickham, Cook & Hofmann 2015). We use principal component analysis (PCA) to map each time series as a point in a two-dimensional instance space given by features. The corresponding results for yearly, monthly and hourly are shown in **??**, **??** and **??**. Most of the yearly series that are classified with very high probability belong to either ARIMA class or random walk with drift class. Distribution of the trend in the feature space shows they are highly trended. Furthermore, these ARIMA series and random walk with drift are even visually distinguishable based on the features beta and diff1y_acf5. Furthermore, the monthly series classified into SARIMA and stlar class with high probability take very low value for entropy, which means the series are easily forecastable. On the other hand the series that are classified in to neural network class with high probability are less forecastable, less trended (low values for beta) and less seasonal (low value for sediff_acf5). For hourly series, the series that are classified with very high probability are

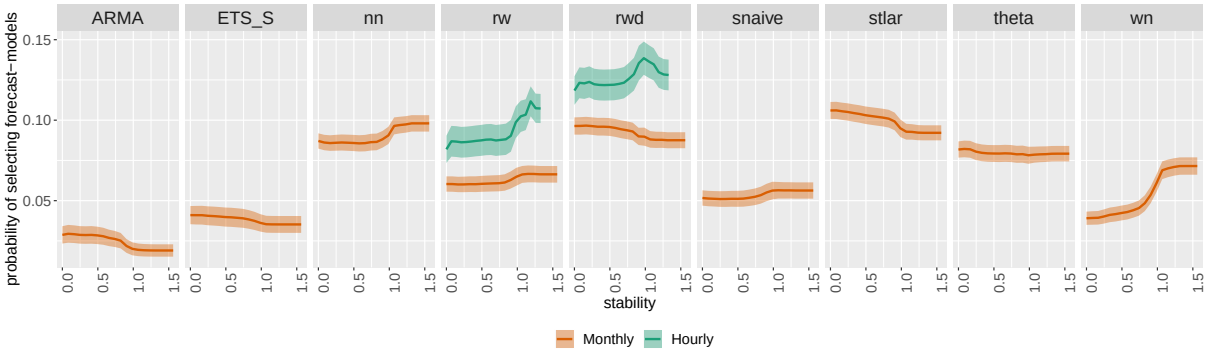

Figure 12: *Partial dependence plots for stability for monthly and hourly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

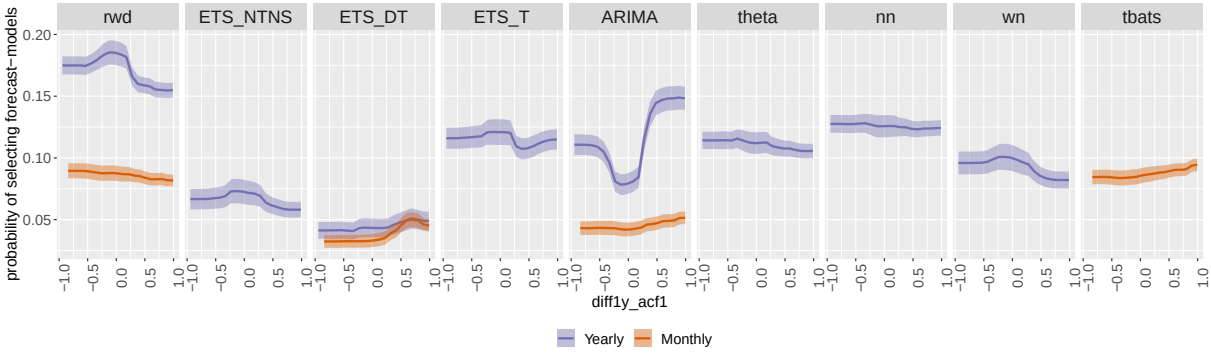

Figure 13: *Partial dependence plots for diff1y_acf1 for yearly and monthly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

very high in length. Lengthy time series means more information, hence easy to recognise clear patterns and therefore easy to classify.

6 Discussion and conclusions

In this paper we have proposed a novel framework for forecast-model selection using meta-learning based on time series features. Our proposed FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify suitable forecast models for a new series. A key advantage of the FFORMS is the idea of outsourcing most of the heavy computational work to the offline phase. Therefore, unlike traditional model selection strategies or cross-validation processes, our proposed framework can be used to accurately forecast very large collections of time series requiring almost instant forecasts. For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the FFORMS random forest classifier, and the calculation of the forecasts from the chosen model. None of these steps involve substantial computation, and they can be easily parallelised when forecasting for a large number of new time series. In

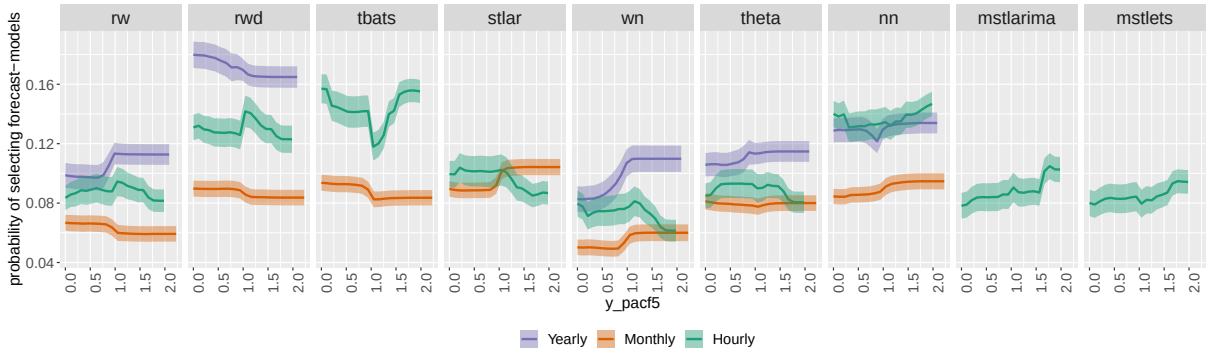

Figure 14: *Partial dependence plots for y_pacf5 for yearly monthly and hourly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

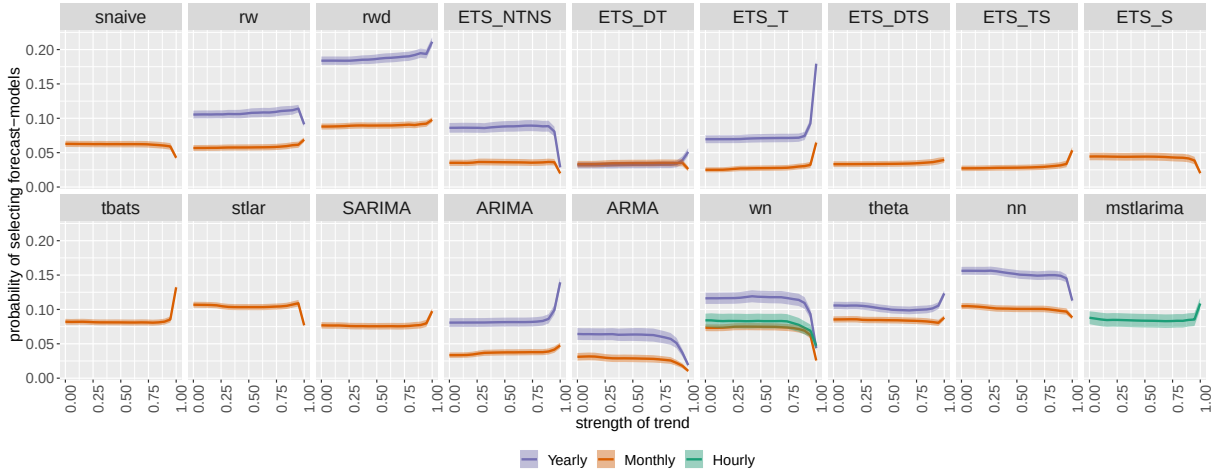

Figure 15: *Partial dependence plots for trend for yearly and monthly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting ETS models without a trend component and stationary models (WN and ARMA) decreases for an extremely high value of trend.*

doing so, the FFORMS framework fills an important gap in contemporary forecasting practice, with many available models to choose from and with predictions being required extremely fast.

This paper makes a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques. Features such as strength of trend, strength of seasonality, linearity, information related to partial-acutorrelation structure, features related to difference stationary, length and entropy are the most important features. However, several features are used to build the framework with comparable contributions, and thus all individual contributions are very small. For all features, the displayed relationships of partial dependency plots are consistent with the domain knowledge expectations. This is an important aspect in encouraging people to trust and use the proposed framework effectively. Further, the results of this study are useful in identifying new ways to improve forecasting accuracy by capturing different features of time

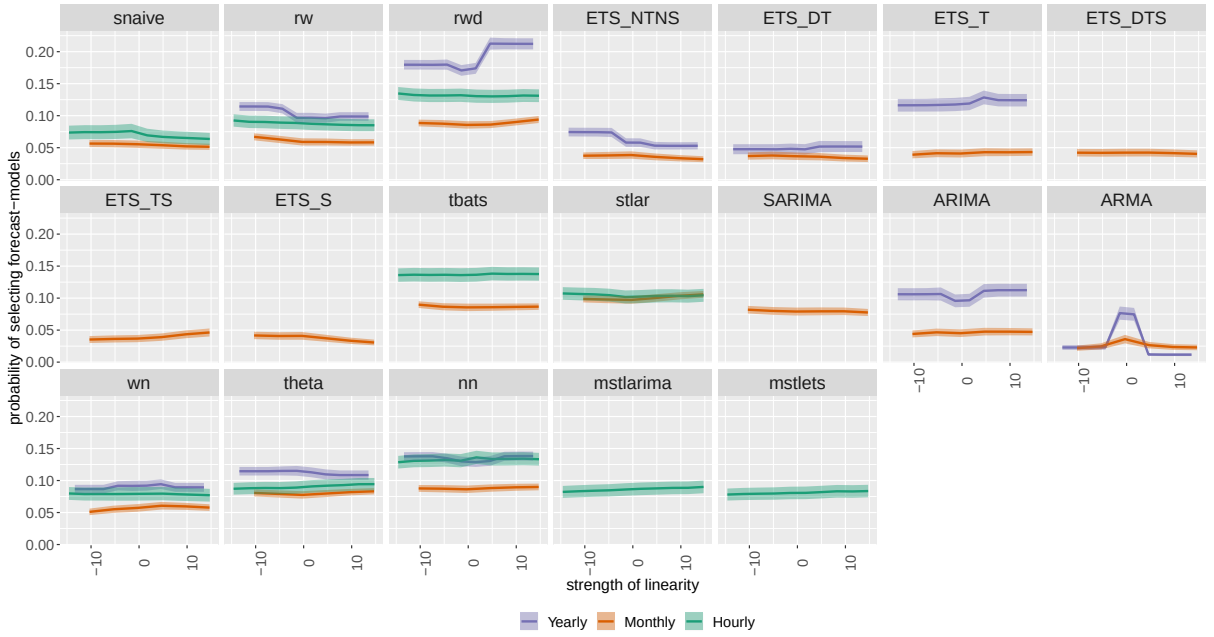

Figure 16: *Partial dependence plots for linearity for yearly monthly and hourly series. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

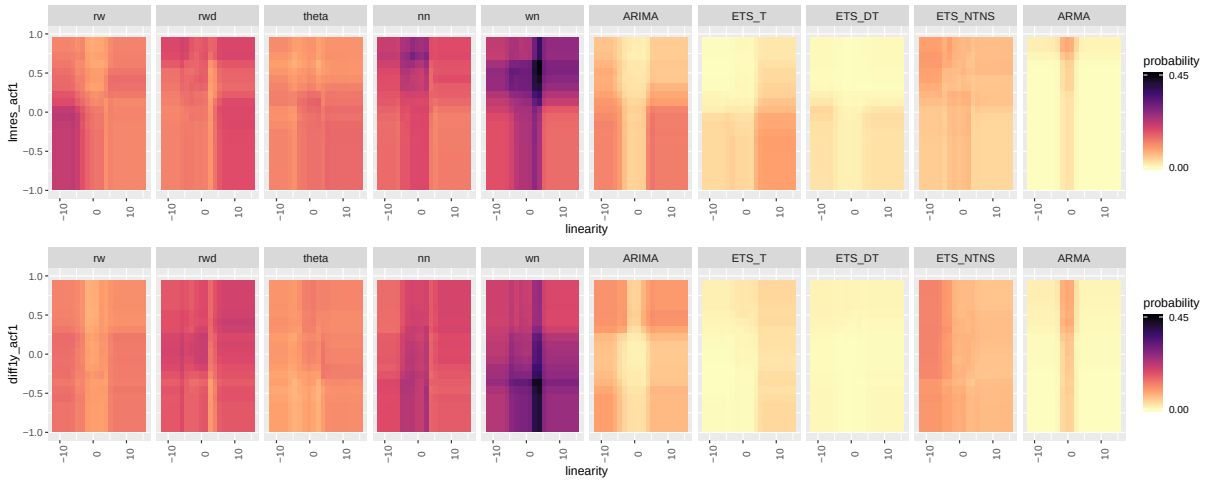

Figure 17: *Two-way partial dependence plots for linearity for yearly series. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

series.

Although we have illustrated the method using the M1, M3 and M4 competition data, the framework is general and can be applied to any large collection of time series. However, the proposed frameworks with the same set of features and forecast models might not be the right choice for forecasting stock return data, intermittent time series or irregular time series, etc. Hence, it is important to expand the frameworks to other datasets that come from different application domains. When adapting the frameworks to other applications the feature space should be revised with appropriate features that measure characteristics of interest. The

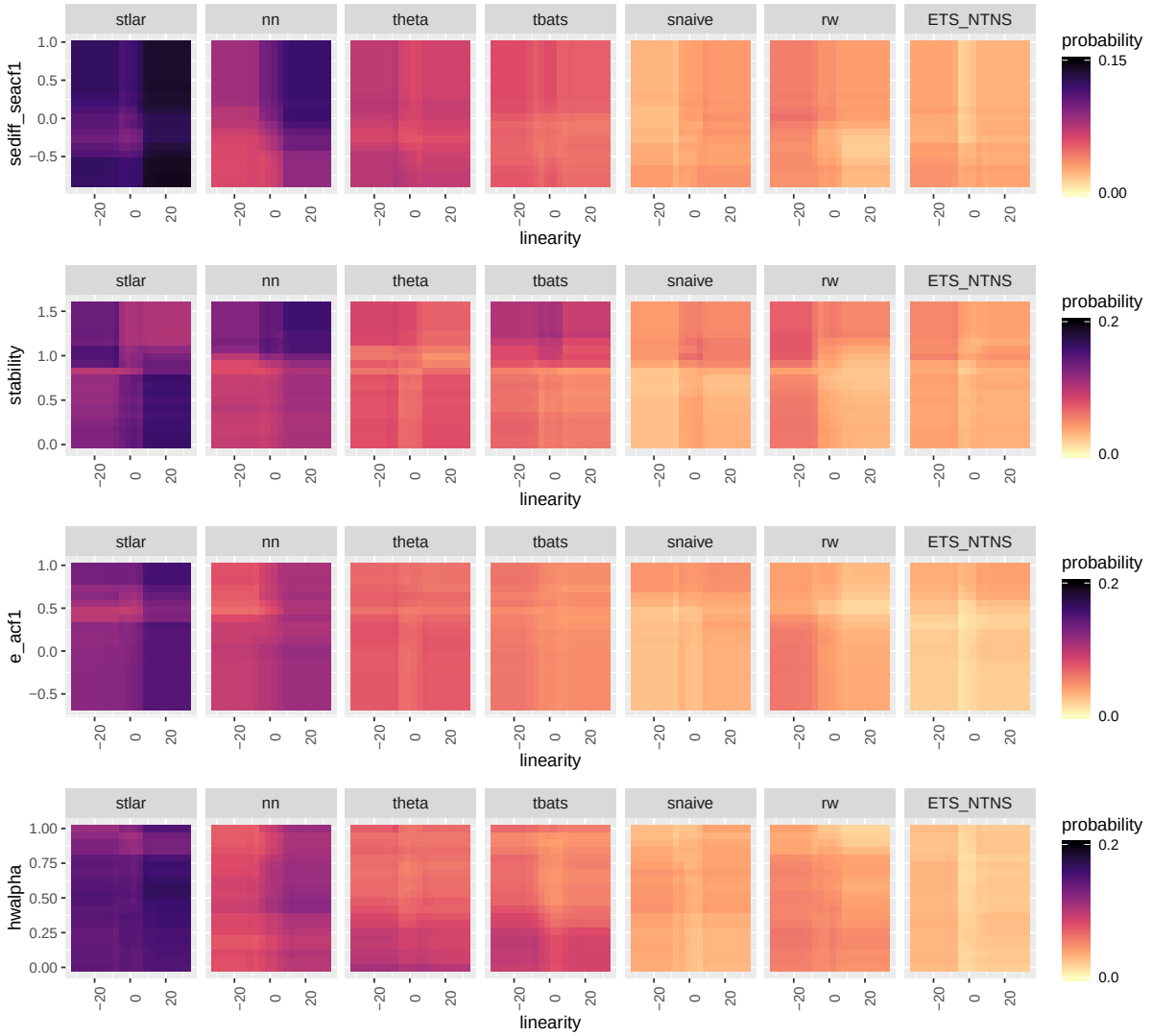

Figure 18: *Two-way partial dependence plots for linearity for monthly series. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

algorithm space should also be revised with suitable forecast models. A suitable forecast error metric should also be considered to evaluate the performance of different forecast models.

Supplemental Materials

Reproducibility: All the code and data used in this paper is available at https://github.com/thiyangt/fforms. The R packages ggplot2 (Hadley 2009) provides the basis for graphics. The document is written in knitr (Xie 2017).

Software: R-package `seer` consists of the implementation of FFORMS algorithm is available on CRAN (https://cran.r-project.org/web/packages/seer/index.html).

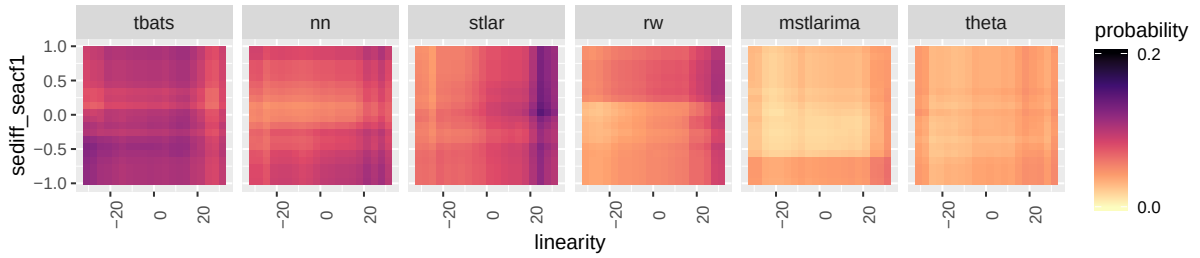

Figure 19: *Two-way partial dependence plots for linearity for hourly series. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

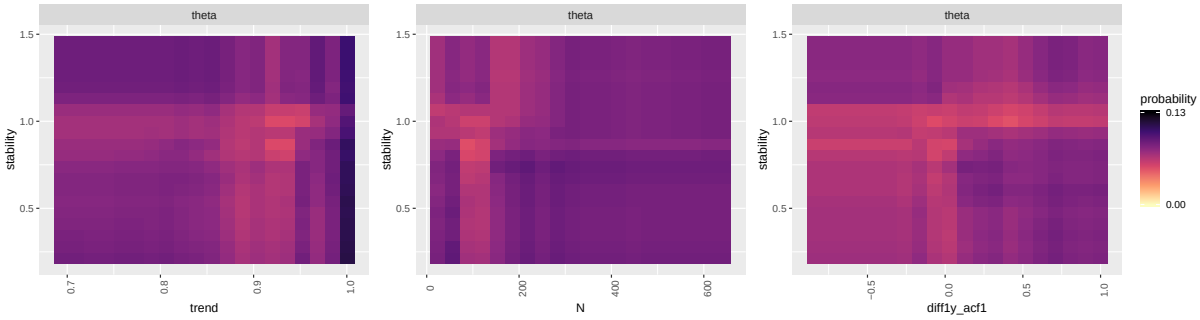

Figure 20: *Two-way partial dependence plots for linearity for monthly series within theta class. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

Appendix A: Definition of features

Length of time series

The appropriate forecast method for a time series depends on how many observations are available. For example, shorter series tend to need simple models such as a random walk. On the other hand, for longer time series, we have enough information to be able to estimate a number of parameters. For even longer series (over 200 observations), models with time-varying parameters give good forecasts as they help to capture the changes of the model over time.

Features based on an STL-decomposition

The strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series, are calculated based on a decomposition of the time series. Suppose we denote our time series as $y_1, y_2, \dots, y_T$. First, an automated Box-Cox transformation (Guerrero 1993) is applied to the time series in order to stabilize the variance and to make the seasonal effect additive. The transformed series is denoted by y_t^*. For quarterly and monthly data, this is decomposed using STL (Cleveland, Cleveland & Terpenning 1990) to give $y_t^* = T_t + S_t + R_t$, where T_t denotes the trend, S_t denotes the seasonal component, and R_t is the remainder component. For non-seasonal data, Friedman's super smoother (Friedman 1984) is

used to decompose $y_t^* = T_t + R_t$, and $S_t = 0$ for all t. The de-trended series is $y_t^* - T_t = S_t + R_t$, the deseasonalized series is $y_t^* - S_t = T_t + R_t$..

The strength of trend is measured by comparing the variance of the deseasonalized series and the remainder series (Wang, Smith-Miles & Hyndman 2009):

$$\text{Trend} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(T_t + R_t)\right].$$

Similarly, the strength of seasonality is computed as

$$\text{Seasonality} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(S_t + R_t)\right].$$

The linearity and curvature features are based on the coefficients of an orthogonal quadratic regression

$$T_t = \beta_0 + \beta_1\phi_1(t) + \beta_2\phi_2(t) + \varepsilon_t,$$

where $t = 1, 2, \ldots, T$, and ϕ_1 and ϕ_2 are orthogonal polynomials of orders 1 and 2. The estimated value of β_1 is used as a measure of linearity while the estimated value of β_2 is considered as a measure of curvature. These features were used by Hyndman, Wang & Laptev (2015). The linearity and curvature depend on the the scale of the time series. Therefore, the time series are scaled to mean zero and variance one before these two features are computed.

The spikiness feature is useful when a time series is affected by occasional outliers. Hyndman, Wang & Laptev (2015) introduced an index to measure spikiness, computed as the variance of the leave-one-out variances of r_t. We compute the first autocorrelation coefficient of the remainder series, r_t.

Stability and lumpiness

The features "stability" and "lumpiness" are calculated based on tiled windows (i.e., they do not overlap). For each window, the sample mean and variance are calculated. The stability feature is calculated as the variance of the means, while lumpiness is the variance of the variances. These were first used by Hyndman, Wang & Laptev (2015).

Spectral entropy of a time series

Spectral entropy is based on information theory, and can be used as a measure of the forecastability of a time series. Series that are easy to forecast should have a small spectral entropy value,

while very noisy series will have a large spectral entropy. We use the measure introduced by Goerg (2013) to estimate the spectral entropy. It estimates the Shannon entropy of the spectral density of the normalized spectral density, given by

$$H_s(y_t) := -\int_{-\pi}^{\pi} \hat{f}_y(\lambda) \log \hat{f}_y(\lambda) d\lambda,$$

where $\hat{f}_y$ denotes the estimate of the spectral density introduced by Nuttall & Carter (1982). The R package ForeCA (Goerg 2016) was used to compute this measure.

Hurst exponent

The Hurst exponent measures the long-term memory of a time series. The Hurst exponent is given by $H = d + 0.5$, where d is the fractal dimension obtained by estimating a ARFIMA$(0, d, 0)$ model. We compute this using the maximum likelihood method (Haslett & Raftery 1989) as implemented in the fracdiff package (Fraley 2012). This measure was also used in Wang, Smith-Miles & Hyndman (2009).

Nonlinearity

To measure the degree of nonlinearity of the time series, we use statistic computed in Terasvirta's neural network test for nonlinearity (Teräsvirta, Lin & Granger 1993), also used in Wang, Smith-Miles & Hyndman (2009). This takes large values when the series is nonlinear, and values around 0 when the series is linear.

Parameter estimates of an ETS model

The ETS (A,A,N) model (Hyndman et al. 2008) produces equivalent forecasts to Holt's linear trend method, and can be expressed as follows:

$$\begin{aligned} y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t, \end{aligned}$$

where α is the smoothing parameter for the level, and β is the smoothing parameter for the trend. We include the parameter estimates of both α and β in our feature set for yearly time series. These indicate the variability in the level and slope of the time series.

The ETS (A,A,A) model (Hyndman et al. 2008) produces equivalent forecasts to Holt-Winters'

additive method, and can be expressed as follows:

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t, \\
s_t &= s_{t-m} + \gamma\varepsilon_t,
\end{aligned}
$$

where γ is the smoothing parameter for the seasonal component, and the other parameters are as above. We include the parameter estimates of α, β and γ in our feature set for monthly and quarterly time series. The value of γ provides a measure for the variability of the seasonality of a time series.

Unit root test statistics

The Phillips-Perron test is based on the regression $y_t = c + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual "Z-alpha" statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.pp()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

The KPSS test is based on the regression $y_t = c + \delta t + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual KPSS statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.kpss()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

Autocorrelation coefficient based features

We calculate the first-order autocorrelation coefficient and the sum of squares of the first five autocorrelation coefficients of the original series, the first-differenced series, the second-differenced series, and the seasonally differenced series (for seasonal data). A linear trend model is fitted to the time series, and the first-order autocorrelation coefficient of the residual series is calculated. We calculate the sum of squares of the first five partial autocorrelation coefficients of the original series, the first-differenced series and the second-differenced series.

References

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

Breiman, L (2001). Random forests. *Machine Learning* **45**(1), 5–32.

Breiman, L & A Cutler (2004). *Random Forests*. Version 5.1. https://www.stat.berkeley.edu/~breiman/RandomForests/.

Breiman, L, A Cutler, A Liaw & M Wiener (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14. https://cran.r-project.org/web/packages/randomForest/.

Chen, C, A Liaw & L Breiman (2004). *Using random forest to learn imbalanced data*. Tech. rep. University of California, Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Cleveland, RB, WS Cleveland & I Terpenning (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Fraley, C (2012). *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. https://CRAN.R-project.org/package=fracdiff.

Friedman, JH (1984). *A variable span scatterplot smoother*. Technical Report 5. Laboratory for Computational Statistics, Stanford University.

Fulcher, BD & NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Goerg, GM (2016). *ForeCA: An R package for forecastable component analysis*. R package version 0.2.4. https://CRAN.R-project.org/package=ForeCA.

Goerg, G (2013). Forecastable component analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp.64–72.

Guerrero, VM (1993). Time-series analysis supported by power transformations. *Journal of Forecasting* **12**, 37–48.

Hadley, W (2009). ggplot2: Elegant graphics for data analysis. *New York: Springer*.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics* **38**(1), 1–50.

Hyndman, R, G Athanasopoulos, C Bergmeir, G Caceres, L Chhay, M O'Hara-Wild, F Petropoulos, S Razbash, E Wang & F Yasmeen (2018). *forecast: Forecasting functions for time series and linear models*. R package version 8.3. http://pkg.robjhyndman.com/forecast.

Hyndman, RJ (2001). It's time to move from what to why. *International Journal of Forecasting* **17**(1), 567–570.

Hyndman, RJ & G Athanasopoulos (2018). *Forecasting: principles and practice*. 2nd ed. Melbourne, Australia: OTexts. https://OTexts.org/fpp2/.

Hyndman, RJ & Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22. http://www.jstatsoft.org/article/view/v027i03.

Hyndman, RJ, AB Koehler, JK Ord & RD Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer.

Hyndman, RJ, AB Koehler, RD Snyder & S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

Kalousis, A & T Theoharis (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* **3**(5), 319–337.

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pp.1499–1506.

Lawrence, M (2001). Why another study? *International Journal of Forecasting* **17**(1), 574–575.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Liaw, A & M Wiener (2002). Classification and regression by randomForest. *R News* **2**(3), 18–22. http://CRAN.R-project.org/doc/Rnews/.

M4 Competitor's Guide (2018). https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf. Accessed: 2018-09-26.

Makridakis, S, A Andersen, R Carbone, R Fildes, M Hibon, R Lewandowski, J Newton, E Parzen & R Winkler (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting* **1**(2), 111–153.

Makridakis, S & M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Makridakis, S, E Spiliotis & V Assimakopoulos (2019). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*.

Nuttall, AH & GC Carter (1982). Spectral estimation using combined time and lag weighting. *Proceedings of the IEEE* **70**(9), 1115–1125.

Pfaff, B (2008). *Analysis of integrated and cointegrated time series with R*. Springer.

Pfaff, B, E Zivot & M Stigler (2016). *urca: Unit root and cointegration tests for time series data*. R package version 1.3-0. https://CRAN.R-project.org/package=urca.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Racine, J (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of Econometrics* **99**(1), 39–61.

Reid, DJ (1972). "A comparison of forecasting techniques on economic time series". In: *Forecasting in Action*. Birmingham, UK: Operational Research Society.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Schep, AN & SK Kummerfeld (2017). iheatmapr: Interactive complex heatmaps in R. *Journal of Open Source Software*. http://dx.doi.org/10.21105/joss.00359.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Smith-Miles, K (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* **41**(1), 6.

Teräsvirta, T, CF Lin & CWJ Granger (1993). Power of the neural network linearity test. *Journal of Time Series Analysis* **14** (2), 209–220.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Wickham, H, D Cook & H Hofmann (2015). Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **8**(4), 203–225.

Widodo, A & I Budi (2013). "Model selection using dimensionality reduction of time series characteristics". Paper presented at the International Symposium on Forecasting, Seoul, South Korea. June 2013. https://goo.gl/ig2J57.

Xie, Y (2017). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC.